SoftwareX 11 (2020) 100474

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

SCOUT: Signal Correction and Uncertainty Quantification Toolbox in MATLAB

Richard Semaan *, Vikas Yadav

Technische Universität Braunschweig, Institute of Fluid Mechanics, Hermann-Blenk-Str. 37, 38106, Braunschweig, Germany

ARTICLE INFO

Article history:
Received 28 November 2019
Received in revised form 2 April 2020
Accepted 2 April 2020

Keywords:
Stationarity
Spurious sample detection
Filtering
Uncertainty analysis
MATLAB

ABSTRACT

This manuscript describes the software package SCOUT, which analyzes, characterizes, and corrects one-dimensional random signals. Specifically, it allows to check and correct for stationarity, detect spurious samples, check for normality, check for periodicity, filter, perform spectral analysis, determine the integral time scale, and perform uncertainty analysis on individual and on propagated signals. The novelty of SCOUT lies in combining these various methods into one compact and easy-to-use toolbox, which enables students and professionals alike to analyze, characterize, and correct for signals without expert knowledge. The program is oriented towards time traces, but an easy adaptation to spatial distributions can be performed by the user. SCOUT is available in two variants: a graphical user interface (GUI) and a script-based version.

© 2020 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

Code metadata

C1	Current code version	v1.0
C2	Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX_2019_366
C3	Code Ocean compute capsule	NA
C4	Legal Code License	BSD-3-Clauses
C5	Code versioning system used	none
C6	Software code languages, tools, and services used	MATLAB
C7	Compilation requirements, operating environments & dependencies	MATLAB Signal Processing Toolbox
C8	If available Link to developer documentation/manual	https://github.com/rsemaan/SCOUT/blob/master/docs/2019_SCOUT_v1.0_UserManual.pdf
C9	Support email for questions	r.semaan@tu-bs.de

1. Motivation and significance

Signal processing and uncertainty quantification are two very broad yet related research areas. They have applications in many fields ranging from engineering to mathematics, to music. With the proper tools, signal analysis allows us to discover valuable traits and characteristics in signals, whereas uncertainties quantification informs us about the origin, propagation, and interplay of different sources of errors. Due to their relevance, many open-source and commercial software packages have been developed, such as the SciPy [1] and the Signal Processing Toolbox [2] packages for signal analysis, and UQLab [3] and propagate [4] packages for uncertainty propagation. These packages, and others, offer an impressive range of capabilities and options but are mainly geared toward expert users in either field. This limits their usability in many engineering fields.

In this manuscript, we present SCOUT, an easy-to-use signal processing and uncertainty quantification MATLAB package that is well suited to today's students and professionals alike. It offers the main tools necessary to analyze, categorize, and quantify the uncertainty of acquired one-dimensional random signals with (possibly) broadband spectrum, as often encountered in fluid flows and speech analysis. We denote a one-dimensional signal one that is dependent on only one variable, such as $s(t)$. In contrast, pictures and videos are expressed as $s(\boldsymbol{x}, \boldsymbol{y})$ and $s(\boldsymbol{x}, \boldsymbol{y}, t)$, respectively, where $\boldsymbol{x}$ and $\boldsymbol{y}$ are spatial coordinates. SCOUT allows to check and correct for stationarity, detect spurious samples,

* Corresponding author.
E-mail address: r.semaan@tu-braunschweig.de (R. Semaan).

https://doi.org/10.1016/j.softx.2020.100474
2352-7110/© 2020 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

Fig. 1. The general layout of SCOUT GUI. The red boxes highlight the various sections of the interface. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

check for normality, check for periodicity, filter, perform spectral analysis, determine the integral time scale, and perform uncertainty analysis on individual and on propagated signals through a data reduction equation. The uncertainty analysis yields uncertainties on central moment up to the fourth moment. SCOUT is available in two variants: a graphical user interface (GUI) version we label SCOUT GUI, and a script-based version we call SCOUT Script. The GUI version offers maximum flexibility to adaptively and visually adjust the analysis settings, whereas the script version enables large batch processing capabilities and own code-integration. The package includes both variants as well as three example scripts with their corresponding signals.

SCOUT differentiates itself from other existing software packages through its combination of signal analysis and uncertainty quantification capabilities, its ease of use, its novel and open GUI- and script-based flexible structure, and its possible broad adaptation in a range of engineering and scientific fields.

2. Software description

The two SCOUT versions, the graphical user interface (GUI) and the script versions are intended to be complimentary. SCOUT GUI provides visual as well as numerical output at every step. It allows dynamic and adaptive analysis with a range of options and parameter tuning. On the other hand, SCOUT Script is command-based designed for integrated and batch processing. In other words, it allows all of SCOUT's capabilities to be integrated within the user's own analysis code. In addition, the script version enables batch processing of large data. An envisioned workflow scenario would be to perform initial analysis using SCOUT GUI, where the settings are fine-tuned, followed by the SCOUT Script for integrated or batch processing.

2.1. Software interface

The two SCOUT versions have two very different interfaces. While the GUI version is adaptive, the script version is static and requires a configuration file. This section details both interfaces.

2.1.1. SCOUT GUI interface

A screenshot of SCOUT GUI's interface is presented in Fig. 1. The red boxes highlight the various regions of the layout:

- Analysis selection: allows selection of analysis type.
- Input: requires the user to input the information necessary for each process.
- Summary: displays the latest general summary of the signal analyzed.
- Plot and plot checkbox: for plotting the various results. and toggling between them.
- Results display: 'announces' important results after the completion of a certain analysis step.
- Signal statistics: displays the latest statistical information of the signal.

The layout structure is general and applies to most of the analysis steps.

2.1.2. SCOUT script interface

SCOUT Script requires one post-processing configuration file for each imported signal, and a single optional uncertainty analysis configuration file. The configuration files contain the various analysis settings. Hence, unlike SCOUT GUI, all analysis settings are fixed during the execution.

The code execution is performed by calling on the main function `SCOUT_Script` followed by the various configuration files:

```
>>SCOUT_Script(ConfigFile1('U','Workscape','u',
10000), . . .  UncertaintyConfigFile)
```

This code snippet, taken from `Example1.m` of the package, shows how `SCOUT_Script` handles the two types of configuration files:

1. Configuration file type 1 (e.g., `ConfigFile1`): is required for the entire analysis sequence *except* for the uncertainty analysis (the last tab in SCOUT GUI).
2. Configuration file type 2 (e.g., `UncertaintyConfigFile`): to specify the uncertainty analysis settings.

In case uncertainty analysis is not sought, the user can simply withhold the type 2 configuration file. Each type 1 configuration file requires four direct inputs:

1. The name of the output summary file.
2. The location of the signal (`Workspace` or `Directory`).
3. The name of the signal in case the file is loaded from the workspace, or the name of the file including its path in case the signal is loaded from a directory.
4. The sampling frequency.

These input options allow batch processing of different signals with different names, saved at different locations, and sampled at different rates. The reader is referred to `Example3` in the package for a batch processing example script.

Besides the three direct input variables, the script version requires a list of other settings inside the configuration files. The various analyses offered and their corresponding settings are detailed in Section 2.2. The user is also referred to the example scripts in the package and the therein-included comments for further details.

2.2. Software functionalities

All analyses begin with importing signals. Up to 5 one-dimensional signals can be simultaneously imported. In SCOUT GUI, the total number of signals to be imported is provided as an option at the top of the input section. Each signal can be either imported from the drive or MATALB's workspace. The user is required to provide the sampling frequency for each signal. It is important to note that when a file containing several signals, *only* the first read signal will be imported.

After importing the signal(s), the user is offered a range of analysis possibilities. This section briefly presents each analysis step.

2.2.1. Stationarity analysis

This analysis examines and optionally renders signals stationary. The stationarity check is based on the reverse arrangement test. For more details about the method, the reader is referred to the ample literature on the subject [5,6].

Stationary analysis requires only one input; The sample size for the reverse arrangement test, which should be a positive number `1 < M < length(signal)`. The default sample size is 100. The sample size should be chosen such that it is neither too short nor too long with respect to the fundamental period of the signal.

If the signal is not stationary, the options for rendering it stationary become active. Signals can be made stationary using two methods: the MATLAB inbuilt `detrend` function, or the proposed `Polynomial fit` method. MATLAB's `detrend` function simply detects and removes linear trends in the data. The user is referred to MATLAB user manual for details. The polynomial fit method uses sequentially higher polynomial orders up to third order[1] to fit the data, and then automatically chooses the best polynomial order based on a compromise between accuracy (fit error) and complexity (polynomial order). After the detrending process, the reverse arrangement test is again repeated to verify stationarity is achieved.

In SCOUT GUI, the process can be separately performed on all imported signals, with every signal individually selected from the drop-down menu. The main steps of the detrending process are registered and updated in the 'Signal summary' section. Similarly, the 'Signal statistics' section gets updated with the new statistical values from the detrended signal. In SCOUT Script, the stationarity process is controlled through 3 variables in the type 1 configuration file.

[1] Higher order polynomials than the third are not recommended [6].

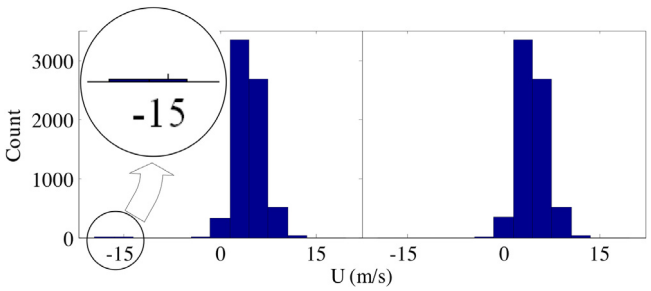


Fig. 2. Illustration of the detection and the elimination of spurious points using the histogram method [9].

2.2.2. Spurious samples

This section describes the detection method and the handling of spurious samples. Two possible algorithms for detecting spurious samples are offered: the Chauvenent criterion, and the histogram method. The Chauvenet criterion [7,8] is a widely accepted method for spurious sample detection. It specifies that all points that fall within $1 - 1/(2N)$ probability band around the mean value should be retained, where N is the sample record length. The Chauvenet criterion can be used on most random data, except when the measured probability density function is skewed or multi-modal, which causes a rejection bias, where 'real' data get clipped. Under this scenario, the so-called histogram approach should be used instead. The histogram approach [9] simply constructs a coarse histogram and detects outliers as samples that are separate from the main histogram body. This approach is illustrated in Fig. 2, where the spurious point in the original data on the left side has been removed, as shown on the right side.

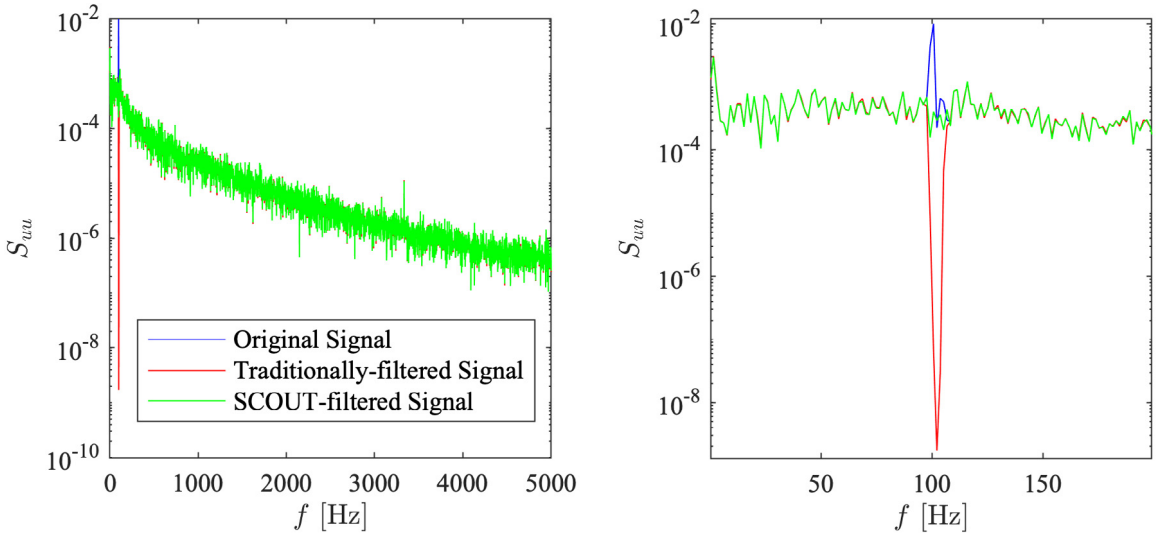
If spurious samples are detected, the options to deal with them become active. In this case, the user has two options: removal, or replacement. As the name suggests, the removal option simply deletes the spurious samples. Removal is the preferred option when no subsequent spectral analysis is planned since any Fourier transform requires data sampled at equal time intervals. Specifically, this will exclude performing 'Periodicity', 'Spectral analysis' and 'Uncertainty analysis'. If any of the above-mentioned analysis is desired, the replacement option should be selected. Here, spurious samples are replaced by their local average values.

In SCOUT GUI, the main steps of the spurious sample detection are visualized in red in the time trace and the histogram plots, and are reported and updated in the 'Signal summary' and the 'Signal statistics' sections. In SCOUT Script, this analysis is controlled with 4 input variables inside the type 1 configuration file.

2.2.3. Normality

This section describes the method to identify normally-distributed signals to a pre-selected confidence level. Among many motivations, knowing whether a signal is Gaussian yields significant simplifications in the uncertainty analysis. The method to check for normality is the χ^2 goodness-of-fit test [10], which is used in a wide range of applications.

In SCOUT GUI, the normality test result is displayed in the 'Result display' section for the selected confidence level. As a visual guide, a normally-distributed reference probability density function with the same mean and standard deviation as the signal is displayed in red in the histogram plot. As before, the main steps of the normality test are registered and updated in the 'Signal summary' section. In SCOUT Script, the normality test is controlled via 2 variables inside the type 1 configuration file.



(a) Entire frequency range.

(b) Zoom-in on the filtered frequency range.

Fig. 3. Comparison between traditional filtering using MATLAB's `bandstop` and SCOUT's filtering method. The spectra of the original signal (blue), of the MATLAB-filtered signal (red), and of the filtered signal using SCOUT (green) are shown. Also shown (b), is a zoom-in on the filtered frequency range. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

2.2.4. Periodicity

This section describes methods to detect periodic components and to optionally filter them. Removing deterministic (e.g. periodic) components from signals is a necessary prerequisite before uncertainty quantification. The periodicity analysis starts with computing the autospectral density function using the Welch's method [11], which takes advantage of block averaging, zero padding, and window overlapping. The magnitude of the highest spectral peak, as well as its frequency, are then detected.

The user is subsequently offered the option to filter the detected (or any other) frequency band using a specially-modified Fourier filter. Unlike typical filters, such as Butterworth or Chebychev, which require a lot of tweaking and tend to over-attenuate the spectrum in the target frequency band yielding a distorted signal in the time domain, the modified Fourier filter is very easy to set and yield a smooth spectrum. The filter simply attenuates the desired frequency range by replacing the Fourier coefficients at those frequencies with a uniformly-distributed *random noise* that have a user-selected standard deviation. The addition of random noise acts as an alternative to windowing [12], and minimizes ripple effects. The attenuation magnitude ranges between level 0, where the Fourier coefficients are replaced with random noise with low standard deviation, and level 1, where the Fourier coefficients are replaced with random noise whose standard deviation is similar to the detected peak magnitude. A comparison between a traditional and the proposed filtering technique is presented in Fig. 3, where the spectra of the original signal (blue), of the MATLAB-filtered signal (red), and of the filtered signal using SCOUT (green) are shown. A zoom-in on the filtered frequency range in Fig. 3(b) clearly shows the over-attenuation when using MATLAB's `bandstop` despite setting the steepness and the attenuation to the very low levels of 0.5 and 10, respectively.

Despite windowing and overlapping, spectral edge-effects are sometimes unavoidable. This issue becomes particularly clear when transforming the signal back to the time domain. SCOUT addresses this issue, by offering the user to append points at both ends of the sample record *before* filtering is initiated [13].

In SCOUT GUI, the detected peak frequency is registered in the 'Signal summary' section. When applicable, the 'Signal statistics' section gets also updated with the new statistics of the filtered signal. In SCOUT Script the spectrum computation and the filtering process is controlled by 8 variables inside the type 1 configuration file.

2.2.5. Spectral analysis

In this section, we detail the spectral analysis that consists of computing the autospectral density function and the autocorrelation coefficient, and estimating the integral time scale. The integral time scale is relevant for physical insights, and for identifying the number of independent samples, which are necessary for the uncertainty analysis. The integral time scale is defined as

$$I = \int_0^{\tau_{\max}} \rho(\tau) d\tau \,, \tag{1}$$

where $\tau_{\max}$ is the maximum time lag, and ρ is the autocorrelation coefficient,

$$\rho = \frac{R_{xx}}{s^2} \,, \tag{2}$$

where R_{xx} is the autocorrelation, and s is the standard deviation. In SCOUT, R_{xx} is computed with an indirect approach from the inverse Fourier transform of the autospectral density function [6]. This approach is computationally faster and makes use of block averaging, delivering a smoother autocorrelation distribution.

Executing the analysis in SCOUT GUI computes and visualizes the auto-spectral density and the autocorrelation coefficient function. The integral time scale value is displayed on the right of the autocorrelation coefficient plot alongside the maximum time lag $\tau_{\max}$. The final computed integral time scale is registered in the 'Signal summary' section. In SCOUT Script, the options for computing the autospectral density function are the same as those listed for the 'Periodicity' analysis' in Section 2.2.4 (`I.Spectral.Window`, `I.Spectral.NumberOfBlocks`, `I.Spectral.Overlap`, `I.Spectral.DiscreteFFT`). Two additional options are required for computing the autocorrelation coefficient and the integral time scale.

2.2.6. Uncertainty analysis

This section describes the uncertainty analysis of individual and of propagated signals. The analysis yields uncertainty quantification of the mean result and of higher-order central moments

up to the fourth. Due to its ease-of-use and direct interpretability, the propagation of uncertainty is performed using the first-order Taylor series method [14,15]. The uncertainty is propagated through a data reduction equation (DRE) of the form

$$r = r(\text{Signal1}, \text{Signal2}, \ldots, \text{Signal5}),$$

where r is the result, and Signal1, ..., Signal5 are the uncertain depend variables. To keep the analysis simple and approachable, the uncertainty propagation analysis is limited to the most practical and recommended approaches:

1. The random uncertainty is computed directly on the result r, as recommended [6]. In other words, individual random uncertainties are not propagated through the DRE, thus bypassing the need to estimate possible correlations among them. This, however, requires that all dependent signals are sampled at (or decimated to) the same sample rate and record length.
2. The systematic uncertainty is only propagated through one accepted functional form of the DRE,

   $$r = C(\text{Signal1})^{n1}(\text{Signal2})^{n2}(\text{Signal3})^{n3},$$

   where C and the n's are user-defined constants.
3. A large sample record is assumed, which yields a coverage factor $t = 2$, i.e., an expanded uncertainty $U_r = 2\,u_r$, where u_r is the combined uncertainty of the result.
4. All reported uncertainties are for 95% confidence level, which is typical for engineering applications.

In SCOUT, the random uncertainty of higher-order central moments can be computed using two approaches. The first one employs simplified equations for the second and fourth central moments,

$$u_{x^2} = \sqrt{\frac{2}{N_{eff}}}, \tag{3a}$$

$$u_{x^4} = \sqrt{\frac{11}{N_{eff}}}, \tag{3b}$$

which assumes a normally-distributed signal. N_{eff} refers to the effective *statistically independent* number of samples. Alternatively, the uncertainties of all central moments can be estimated directly with

$$u_{x^m} = \sqrt{\frac{1}{N_{eff}} \frac{\langle x^{2\,m}\rangle - \langle x^m\rangle^2}{\langle x^m\rangle^2}}, \tag{4}$$

where $\langle\ \ \rangle$ is the time-averaging operator. Direct equation (4) is only recommended when the signal(s) is (are) sampled for a sufficiently long time.

Executing the uncertainty analysis in SCOUT GUI yields two plots and various outputs, such as the systematic uncertainty of the result, and uncertainties of the second, third and fourth central moments. The entire uncertainty analysis in SCOUT Script is set with one type 2 configuration file.

3. Impact

Signal processing and uncertainty quantification and propagation are fundamental requirements for the engineering sciences. However, the wealth of information and the complexity of methods are hindering broad adaptation. Thus, a compact, interactive, and easy-to-use toolbox provides an attractive alternative to existing expansive packages, which typically require expert knowledge. The price to pay is the limited offered options for signal processing and for uncertainty analysis, which can be easily remedied through own-code integration. With its two flavors, SCOUT offers unique capabilities for both interactive and integrated analysis.

The engineering sciences offer plentiful application possibilities to the more theoretical fields. However, the spectrum of skills required by engineers is getting broader every day. For example, in the field of fluid mechanics, an experimentalist is typically required to possess skills in flow physics, mechanical design, measurement techniques, programming, data analysis, and uncertainty quantification. SCOUT helps alleviate the burden by providing the necessary tools throughout the various stages of an experiment; It can be used before the start of the experiment to initially assess the required measurement accuracy of the acquisition systems for a target result uncertainty. SCOUT can be employed during the experiment for quick detection of experimental anomalies, such as drifts and signal dropouts. After the conclusion of the experiment, the experimentator can make use of SCOUT's signal processing and uncertainty quantification tools.

4. Conclusions

Signal Correction and Uncertainty Quantification Toolbox (SCOUT) is a user-friendly MATLAB package for signal analysis. It builds on years of experience and best practices in processing experimental fluid flow data. It covers a range of analyses typically encountered when processing measured signals. These include stationarity analysis, spurious samples detection, normality check, periodicity check, filtering, spectral analysis, and uncertainty analysis of individual and of multiple propagated signals through a data reduction equation. Numerical checks show the consistency and validity of the results.

SCOUT is an ongoing project. The authors are committed to its future development, which include more spectral analysis options, such as wavelet transform, and expanding the uncertainty propagation analysis to include more general functional forms and direct computation of the gradient for the Taylor series expansion method.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We acknowledge support by the German Research Foundation and the Open Access Publication Funds of the Technische Universität Braunschweig.

References

[1] Jones E, Oliphant T, Peterson P, et al. SciPy: Open source scientific tools for Python. 2001–2019, URL http://www.scipy.org/.
[2] Mathworks. MATLAB signal processing toolbox. 2019, URL http://www.mathworks.com/products/signal.html.
[3] Marelli S, Sudret B. UQLab: A framework for uncertainty quantification in MATLAB, In: Proceedings in the 2nd international conference on vulnerability, risk analysis and managemen. Liverpool, United Kingdom. 2014. p. 2554–2563. http://dx.doi.org/10.1061/9780784413609.257.
[4] Spiess A-N. Propagate: Propagation of uncertainty in R. 2018, URL https://CRAN.R-project.org/package=propagate.
[5] Siegel S, Castellan NJ. Nonparametric statistics for the behavioral sciences. 2nd ed.. McGraw-Hill; 1988.
[6] Bendat JS, Piersol AG. Random data: analysis and measurement procedures. 4th ed.. John Wiley & Sons; 2011.
[7] Chauvenet W. A manual of spherical and practical astronomy, Vol. II. 5th ed.. New York: Dover Publication; 1960 (reprinted).

[8] Coleman HW, Steele GW. Experimentation, validation, and uncertainty analysis for engineers. 3rd ed.. John Wiley & Sons; 2009.
[9] Semaan R, Naughton JW. Three-component laser-Doppler-anemometry measurements in turbulent swirling jets. AIAA J. 2013;51(9):2098–113. http://dx.doi.org/10.2514/1.J051783.
[10] Cochran WG. The χ^2 test of goodness of fit. Ann. Math. Stat. 1952;23(3):315–45.
[11] Welch P. The use of fast Fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms. IEEE Trans. Audio Electroacoust. 1967;15(2):70–3.
[12] Kemao Q. Applications of windowed Fourier fringe analysis in optical measurement: a review. Opt Lasers Eng 2015;66:67–73. http://dx.doi.org/10.1016/j.optlaseng.2014.08.012.
[13] Mayhew MA. Curie isotherm surfaces inferred from high-altitude magnetic anomaly data. J Geophys Res Solid Earth 1985;90(B3):2647–54. http://dx.doi.org/10.1029/JB090iB03p02647.
[14] Tukey JW. The propagation of errors, fluctuations, and tolerances basic generalized formulas. Tech. Rep., (10). N.J.: Statistical Techniques Research Group, Princeton University; 1957.
[15] C.H.elton J, Davis FJ. Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems. Reliab. Eng. Syst. Saf. 2003;81(1):23–69. http://dx.doi.org/10.1016/S0951-8320(03)00058-9.